Scalable importance sampling estimation of Gaussian mixture posteriors in Bayesian networks

Darío Ramos-López[a,*], Andrés R. Masegosa[a], Antonio Salmerón[a], Rafael Rumí[a], Helge Langseth[c], Thomas D. Nielsen[b,a], Anders L. Madsen[d,b]

[a] *Department of Mathematics, University of Almería, Spain*
[b] *Department of Computer Science, Aalborg University, Denmark*
[c] *Department of Computer and Information Science, The Norwegian University of Science and Technology, Norway*
[d] *HUGIN EXPERT A/S, Aalborg, Denmark*

Abstract

In this paper we propose a scalable importance sampling algorithm for computing Gaussian mixture posteriors in conditional linear Gaussian Bayesian networks. Our contribution is based on using a stochastic gradient ascent procedure taking as input a stream of importance sampling weights, so that a mixture of Gaussians is dynamically updated with no need to store the full sample. The algorithm has been designed following a Map/Reduce approach and is therefore scalable with respect to computing resources. The implementation of the proposed algorithm is available as part of the AMIDST open-source toolbox for scalable probabilistic machine learning (`http://www.amidsttoolbox.com`).

Keywords: Importance sampling, Bayesian networks, Conditional Linear Gaussian models, Scalable inference, Gaussian mixtures

1. Introduction

Bayesian networks (BNs) (Jensen & Nielsen, 2007; Pearl, 1988) provide a well-founded and principled approach for Bayesian reasoning in complex domains endowed with uncertainty. A prominent feature of BNs as a framework

*Corresponding author
Email address: `dramoslopez@ual.es` (Darío Ramos-López)

for representing uncertain knowledge is the possibility for defining efficient algorithms for performing probabilistic inference (e.g. computation of the posterior distribution of a target variable). These algorithms are typically designed to take advantage of the independence properties implied by the structure of the Bayesian network (Jensen et al., 1990; Madsen, 2010; Madsen & Jensen, 1999; Shenoy, 1997).

Even though most of the methodological development around Bayesian networks has focused on discrete variables, there are plenty of problems in which discrete and continuous variables coexist. A BN is called *hybrid* if it contains discrete and continuous variables simultaneously. The most established approach for explicitly handling hybrid BNs came with the definition of the conditional linear Gaussian (CLG) model (Lauritzen & Wermuth, 1989). This model class is based on the assumption of normality over the continuous variables, and the structural restriction that prevents discrete variables from having continuous parents. If these modeling assumptions are not congruent with the domain being modeled, one may instead opt to discretize the continuous variables (Kozlov & Koller, 1997; Neil et al., 2007, 2008), thus transforming the hybrid BN to a standard discrete BN. Unfortunately, such transformations typically also result in a loss of information.

Mixtures of truncated basis functions (Langseth et al., 2012) provide a generalization of standard discretization and do not impose any structural restriction on the model nor do they make distributional assumptions like the normality assumption imposed by CLG models. Furthermore, they are compatible with exact probabilistic inference algorithms as, for instance, the Shenoy-Shafer architecture (Shenoy & Shafer, 1990) and the variable elimination scheme (Zhang & Poole, 1996). However, the complexity of probabilistic inference in these models often renders them inappropriate when dealing with a large number of variables and a limited response time (Shenoy et al., 2015).

2. Motivation and contribution

In this paper, we are interested in approximate probabilistic inference methods for hybrid BNs satisfying the following two requirements: (i) they should be able to scale with the computational resources available in order to provide results after a short computing time; and (ii) the provided output should be an explicit probability density, rather than just a set of quantiles or moments of the distribution. A widely applicable scenario where both requirements are needed comes up when processing data streams at high speed, and for each item in the stream, we need to know the result of processing the item through probabilistic inference in a hybrid BN. The result of the inference process should be quickly available (before the next data item arrives). At the same time, the availability of the explicit form of the posterior density facilitates subsequent analysis. For instance, as a basis for expected utility calculations or as a tool for anomaly detection from the input data stream.

For the former reason, we focus our analysis on CLG models, instead of less restrictive alternatives such as mixtures of truncated basis functions, as inference in the latter models is in general more time consuming (Rumí & Salmerón, 2007). Another advantage of CLG models is that it is known that the posterior distribution on a continuous variable in a CLG network is always a mixture of Gaussians (Lauritzen & Jensen, 2001).

Many approaches can be used to perform approximate inference in CLG models. Deterministic approximations include (mean field) variational methods (Winn & Bishop, 2005) and expectation propagation (Minka, 2001). The problem is that these methods are iterative in nature and, as a consequence, difficult to parallelize and scale up. Scalable alternatives exist (Hoffman et al., 2013; Masegosa et al., 2017a), but they are not designed for general BNs, but for restricted plate models oriented to learning problems. Furthermore, the approximations provided by these models are often expressed by a single Gaussian distribution in order to make the methods computationally efficient, but, as we will show in the experimental section, this approximation is usually not

sufficiently accurate.

Monte-Carlo methods define another widely used class of approximate inference approaches which could be used in this setting. An important group of them are based on the importance sampling technique, that provides a flexible approach for constructing *anytime* probabilistic reasoning algorithms (Cheng & Druzdzel, 2000; Moral & Salmerón, 2005; Yuan & Druzdzel, 2005, 2007), where the term *anytime* means that the accuracy of the results provided by an algorithm is proportional to the time it is allowed to run (Ramos & Cozman, 2005). The advantage of importance sampling methods is that they are embarrassingly parallelizable, as shown in (Salmerón et al., 2015). However, the plain application of importance sampling yields an empirical distribution that approximates the posterior, rather than an explicit density (a mixture of Gaussians, for instance).

In this paper we extend the method in (Salmerón et al., 2015) enabling it to compute mixture of Gaussians posterior densities. Our contribution is based on using a stochastic gradient ascent procedure taking as input a stream of importance sampling weights, so that a mixture of Gaussians is dynamically updated with no need to store the full sample. The algorithm has been designed following a Map/Reduce approach and is therefore scalable with respect to computing resources. The implementation of the algorithm is available as part of the AMIDST open-source toolbox for scalable probabilistic machine learning (`http://www.amidsttoolbox.com`) (Masegosa et al., 2017b).

3. Preliminaries

Consider a set of N random variables $\mathbf{X} = \{X_1, \ldots, X_N\}$. A BN over $\mathbf{X}$ is composed of a directed acyclic graph, where each node represents a variable in $\mathbf{X}$, and a set of conditional probability distributions such that the joint distribution over $\mathbf{X}$ factorizes as

$$p(\mathbf{X}) = \prod_{i=1}^{N} p_i(X_i|\mathrm{pa}(X_i)), \tag{1}$$

where $\text{pa}(X_i)$ denotes the set of parents of X_i in the graph representation.

We will use lowercase letters to refer to values or configurations of values, so that x denotes a value of X and boldface $\mathbf{x}$ is a configuration of the variables in $\mathbf{X}$. Given a set of observed variables $\mathbf{X}_E \subset \mathbf{X}$ and a set of variables of interest $\mathbf{X}_I \subset \mathbf{X} \setminus \mathbf{X}_E$, *probabilistic inference*, also called *belief update*, consists of computing the posterior distribution $p(x_i|\mathbf{x}_E)$ for each $i \in I$; here we allow X_i to be either discrete or continuous[1].

If we denote by $\mathbf{X}_C$ and $\mathbf{X}_D$ the set of continuous and discrete variables not in $\{X_i\} \cup \mathbf{X}_E$, and by $\mathbf{X}_{C_i}$ and $\mathbf{X}_{D_i}$ the set of continuous and discrete variables not in $\mathbf{X}_E$, the goal of probabilistic inference can generally be formulated as computing

$$p(x_i|\mathbf{x}_E) = \frac{p(x_i, \mathbf{x}_E)}{p(\mathbf{x}_E)} = \frac{\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_C}} p(x_i, \mathbf{x}_C, \mathbf{x}_D, \mathbf{x}_E) \mathrm{d}\mathbf{x}_C}{\sum_{\mathbf{x}_{D_i} \in \Omega_{\mathbf{X}_{D_i}}} \int_{\mathbf{x}_{C_i} \in \Omega_{\mathbf{X}_{C_i}}} p(\mathbf{x}_{C_i}, \mathbf{x}_{D_i}, \mathbf{x}_E) \mathrm{d}\mathbf{x}_{C_i}}, \quad (2)$$

where $\Omega_{\mathbf{X}}$ is the support of a set of variables $\mathbf{X}$ and $p(x_i, \mathbf{x}_C, \mathbf{x}_D, \mathbf{x}_E) = p(\mathbf{x}_{C_i}, \mathbf{x}_{D_i}, \mathbf{x}_E)$ is the joint distribution in the BN instantiated according to the observed values $\mathbf{x}_E$.

If X_i in Equation (2) is continuous, the result of probabilistic inference is the evaluation of a density function. In this case one is typically interested in the probability of the variable taking values in a given interval (a, b). This amounts to computing

$$p(X_i \in (a, b)|\mathbf{x}_E) = \frac{\int_a^b \sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_C}} p(x_i, \mathbf{x}_C, \mathbf{x}_D, \mathbf{x}_E) \mathrm{d}\mathbf{x}_C \mathrm{d}x_i}{\sum_{\mathbf{x}_{D_i} \in \Omega_{\mathbf{X}_{D_i}}} \int_{\mathbf{x}_{C_i} \in \Omega_{\mathbf{X}_{C_i}}} p(\mathbf{x}_{C_i}, \mathbf{x}_{D_i}, \mathbf{x}_E) \mathrm{d}\mathbf{x}_{C_i}}. \quad (3)$$

If X_i is discrete, instead of the variable taking values in an interval, we seek

[1] In this paper we only consider inference wrt. the posterior marginal distribution of a variable and not joint distributions over several variables.

to compute the posterior probability of one of its possible values, i.e.

$$p(X_i = x_i|\mathbf{x}_E) = \frac{\sum_{\mathbf{x}_D \in \Omega_{\mathbf{X}_D}} \int_{\mathbf{x}_C \in \Omega_{\mathbf{X}_C}} p^{R(X_i=x_i)}(x_i, \mathbf{x}_C, \mathbf{x}_D, \mathbf{x}_E) \mathrm{d}\mathbf{x}_C}{\sum_{\mathbf{x}_{D_i} \in \Omega_{\mathbf{X}_{D_i}}} \int_{\mathbf{x}_{C_i} \in \Omega_{\mathbf{X}_{C_i}}} p(\mathbf{x}_{C_i}, \mathbf{x}_{D_i}) \mathrm{d}\mathbf{x}_{C_i}}, \quad (4)$$

where $p^{R(X_i=x_i)}(x_i, \mathbf{x}_{C_i}, \mathbf{x}_{D_i}, \mathbf{x}_E)$ denotes the restriction of $p(x_i, \mathbf{x}_C, \mathbf{x}_D, \mathbf{x}_E)$ to the value x_i of variable X_i. We will generically refer to the probabilistic inference tasks described in Equations (3) and (4) as *queries*.

3.1. Conditional Linear Gaussian Networks

A *CLG network* is a hybrid BN where the joint distribution is a CLG (Lauritzen & Wermuth, 1989). In the CLG model, the conditional distribution of each discrete variable $X_D \in \mathbf{X}$ given its parents is a multinomial, whilst the conditional distribution of each continuous variable $Z \in \mathbf{X}$ with discrete parents $\mathbf{X}_D \subseteq \mathbf{X}$ and continuous parents $\mathbf{X}_C \subseteq \mathbf{X}$, is a normal density defined as

$$p(z|\mathbf{X}_D = \mathbf{x}_D, \mathbf{X}_C = \mathbf{x}_C) = \mathcal{N}(z|\alpha_{\mathbf{x}_D} + \boldsymbol{\beta}^{\mathrm{T}}_{\mathbf{x}_D} \mathbf{x}_C, \sigma_{\mathbf{x}_D}), \quad (5)$$

for all $\mathbf{x}_D \in \Omega_{\mathbf{X}_D}$ and $\mathbf{x}_C \in \Omega_{\mathbf{X}_C}$, where α and $\boldsymbol{\beta}$ are the coefficients of a linear regression model of Z given its continuous parents; this model can differ for each configuration of the discrete variables $\mathbf{X}_D$. Therefore, the conditional mean of Z follows a linear model on its continuous parents, while its standard deviation, σ_D, only depends on the discrete ones.

After instantiating the discrete variables, the joint distribution of any subset $\mathbf{X}_C \subseteq \mathbf{X}$ of continuous variables can be obtained in closed form. More precisely, it is a multivariate Gaussian whose parameters can be obtained from those of the CLG representation. Consider a set of M continuous variables $Z_1, \ldots, Z_M$ with a conditionally specified joint density

$$p(z_1, \ldots, z_M) = \prod_{k=1}^{M} p(z_k|z_{k+1}, \ldots, z_M), \quad (6)$$

and where the k-th factor, $1 \leq k \leq M$, is such that

$$p(z_k|z_{k+1}, \ldots, z_M) = \mathcal{N}(z_k|\mu_{z_k|z_{k+1},\ldots,z_M}, \sigma_{z_k}). \quad (7)$$

For this model it holds that the joint density is

$$p(z_1, \ldots, z_M) = \mathcal{N}(z_1, \ldots, z_M | \boldsymbol{u}, \boldsymbol{\Sigma}), \tag{8}$$

where $\boldsymbol{u}$ is the M-dimensional vector of means and $\boldsymbol{\Sigma}$ is the covariance matrix of the multivariate distribution over random variables $Z_1, \ldots, Z_M$ and both $\boldsymbol{u}$ and $\boldsymbol{\Sigma}$ are derived from the parameters in Equation (5) as described, for instance, in (Shachter & Kenley, 1989).

3.2. Importance Sampling

In this section we analyze an approach to approximate inference in CLG networks based on a general technique for probabilistic inference able to provide quick answers to queries, namely *importance sampling* (Hammersley & Handscomb, 1964).

Assume we have a random variable X with density $p(x)$. Importance sampling is a versatile simulation technique designed for estimating the expected value of a given function, f, of random variable X. It is based on the following transformation:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int \frac{p(x)}{p^*(x)} f(x)p^*(x)dx = \mathbb{E}_{p^*}\left[\frac{p(x)}{p^*(x)} f(x)\right],$$

where p^* is a density function called *the sampling distribution* or *the proposal distribution*, verifying that $p^*(x) > 0$ whenever $p(x) > 0$. Therefore, $\mathbb{E}_p[f(x)]$ can be estimated by drawing a sample $x^{(1)}, \ldots, x^{(m)}$ from p^* and computing

$$\hat{\mathbb{E}}_p[f(x)] = \frac{1}{m} \sum_{j=1}^{m} \frac{p(x^{(j)})}{p^*(x^{(j)})} f(x^{(j)}), \tag{9}$$

which is specially convenient if p^* is easier to handle than p.

Importance sampling can be applied to probabilistic inference in BNs taking into account that we can write $p(X_i \in (a,b)|\mathbf{x}_E)$ in Equation (3) as

$$\begin{aligned} p(X_i \in (a,b)|\mathbf{x}_E) &= \mathbb{E}_{p(x_i|\mathbf{x}_E)}\left[\mathbf{1}_{(a,b)}(x_i)\right] = \mathbb{E}_{p^*}\left[\frac{p(x_i|\mathbf{x}_E)}{p^*(x_i)}\mathbf{1}_{(a,b)}(x_i)\right] \\ &= \frac{1}{p(\mathbf{x}_E)}\mathbb{E}_{p^*}\left[\frac{p(x_i, \mathbf{x}_E)}{p^*(x_i)}\mathbf{1}_{(a,b)}(x_i)\right], \end{aligned}$$

where $\mathbf{1}_{(a,b)}$ is the indicator function on interval (a, b). From now on, we will use the notation $g(x_i) = p(x_i, \mathbf{x}_E)$. Hence,

$$\begin{aligned}\hat{p}(X_i \in (a,b)|\mathbf{x}_E) &= \frac{1}{p(\mathbf{x}_E)}\hat{\mathbb{E}}_{p^*}\left[\frac{g(x_i)}{p^*(x_i)}\mathbf{1}_{(a,b)}(x_i)\right] \\ &= \frac{1}{p(\mathbf{x}_E)}\frac{1}{m}\sum_{j=1}^{m}\frac{g(x_i^{(j)})}{p^*(x_i^{(j)})}\mathbf{1}_{(a,b)}(x_i^{(j)}).\end{aligned}$$

The normalizing constant, $p(\mathbf{x}_E)$, can be estimated using the same sample $x_i^{(1)}, \ldots, x_i^{(m)}$, just by summing all the *weights* (Fernández et al., 2012):

$$\hat{p}(\mathbf{x}_E) = \frac{1}{m}\sum_{j=1}^{m}\frac{g(x_i^{(j)})}{p^*(x_i^{(j)})}.$$

Direct calculations using those weights may lead to numerical instability due to underflow (for instance, on relatively large networks or when the evidence has a low likelihood). These problems can be handled by using the so-called *log-sum-exp trick*. Details are provided in Appendix B.

In general when computing the expectation of a function f of a random variable X_i using the posterior distribution $p(x_i|\mathbf{x}_E)$ we have

$$\hat{\mathbb{E}}_{p(x_i|\mathbf{x}_E)}[f(x_i)] = \frac{1}{p(\mathbf{x}_E)}\hat{\mathbb{E}}_{p^*}\left[\frac{g(x_i)}{p^*(x_i)}f(x_i)\right]. \tag{10}$$

It can be seen that the estimator in Equation (9) is unbiased, and its variance is determined by the proposal distribution. A simple procedure for selecting the proposal distribution is the so-called *evidence weighting* (EW) algorithm (Fung & Chang, 1990). In EW, each variable is sampled from a conditional density given its parents in the network. The sampling order is therefore from parents to children. The observed variables are not sampled, but are instead instantiated to their observed values.

Hence, adopting EW means that p involves the product of all the conditional distributions in the BN, whereas p^* involves the same conditional distributions except those associated with the observed variables. For the sake of simplicity

and scalability, we adopt EW as the underlying sampling algorithm. However, more sophisticated ways of obtaining the sampling distribution can also be incorporated to the proposed algorithms, possibly at the cost of higher computational workload.

4. Posterior density estimation in CLG networks

In this section we describe our proposal for computing posterior distributions in CLG networks. Our goal is to fit a posterior density relying on the importance sampling methodology described in Section 3.2. Our proposed methods perform this estimation in an on-line fashion by updating the posterior distributions after every sample is generated by the importance sampling algorithm, thus keeping the memory usage constant.

Let us denote by $q(x_i|\boldsymbol{\theta})$ the target posterior density which we assume to be parametrized by the vector $\boldsymbol{\theta}$. This posterior density aims to approximate the true posterior, $p(x_i|\boldsymbol{x}_E)$. More precisely, the problem we seek to solve is the following one:

$$\arg\min_{\boldsymbol{\theta}} KL\big(p(x_i|\boldsymbol{x}_E)|q(x_i|\boldsymbol{\theta})\big), \tag{11}$$

where $KL(p||q)$ is the Kullback-Leibler divergence (Kullbach & Leibler, 1951) from p to q.

In other words, we look for the distribution q which is closest in terms of KL divergence to the true posterior distribution. The above problem is similar to the problem solved by variational inference methods, but with the difference that variational methods attempt to minimize the reverse KL divergence, which has strong implications on the nature of the approximations that can be obtained (Blei et al., 2016). The objective in Equation (11) was chosen because of its favourable behaviour when approximating multi-modal and strongly peaked posteriors, see, for instance, Murphy (2012, Ch. 21.2.2).

4.1. Fitting a Gaussian posterior density

In the simple case where we aim to fit a Gaussian distribution with parameters μ and σ^2, we just need to estimate the first and second order moments of the posterior distribution, $\mathbb{E}_{p(x_i|\boldsymbol{x}_E)}[X_i]$ and $\mathbb{E}_{p(x_i|\boldsymbol{x}_E)}[X_i^2]$, which can be estimated as described in Section 3.2 by letting $f(x_i) = x_i$ and $f(x_i) = x_i^2$, respectively. The resulting estimators are

$$\hat{\mu} = \hat{\mathbb{E}}_{p(x_i|\boldsymbol{x}_E)}[X_i]$$

and

$$\hat{\sigma}^2 = \hat{\mathbb{E}}_{p(x_i|\boldsymbol{x}_E)}[X_i^2] - \hat{\mu}^2.$$

The details are given in Algorithm 1, which uses Algorithm 2 for generating samples and importance weights. Notice how the procedure does not need to store the generated sample, as all the information it contains is captured by the sufficient statistics in Steps 5 and 6.

4.2. Online estimation of the posterior using a mixture of Gaussians

Since it is known that the posterior over any continuous variable in a CLG network is a mixture of Gaussians (MoG), fitting a MoG rather than a single Gaussian is likely to provide more accurate posteriors. Formally, a K-component MoG density is represented as follows,

$$q(x|\mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{i=1}^{K} w_i \mathcal{N}(x|\mu_i, \sigma_i), \tag{12}$$

where $\mathbf{w} = \{w_1, \ldots, w_K\}$ are the mixing coefficients, $0 < w_i < 1$, $\sum_{i=1}^{K} w_i = 1$, $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_K\}$ are the means, and $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_K\}$ and standard deviations of the K components.

In this case, the method described above for a single Gaussian distribution cannot be applied. To address this problem we propose an online algorithm inspired by stochastic extensions of the EM algorithm (Masegosa, 2014). The proposed algorithm is able to operate over the importance weighted samples in a streaming fashion, and has the same computational complexity as Algorithm 1

Function EW_normal($\mathbf{X}$, P, $\mathbf{x}_E$, X_i, M)

Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$ in topological order. The distributions in the network $P = \{p_1, \ldots, p_N\}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$. The target variable X_i. Sample size M.

Output: An estimation of the mean and variance of $p(x_i|\mathbf{x}_E)$.

begin
  $s_1 \leftarrow 0$; $s_2 \leftarrow 0$; $sumW \leftarrow 0$
  for $j \leftarrow 1$ **to** M **do**
    $(x_i^{(j)}, W) \leftarrow$ EW_Simulate($\mathbf{X}$, P, $\mathbf{x}_E$, X_i) (see Algorithm 2)
    $s_1 \leftarrow s_1 + x_i^{(j)} \cdot W$
    $s_2 \leftarrow s_2 + \left(x_i^{(j)}\right)^2 \cdot W$
    $sumW \leftarrow sumW + W$
  end
  $\mu \leftarrow s_1/sumW$
  $\sigma^2 \leftarrow s_2/sumW - \mu^2$
  return μ, σ^2 *(or s_1, s_2, $sumW$).*
end

Algorithm 1: The EW algorithm for estimating the mean and variance of the posterior distribution.

Function EW_Simulate($\mathbf{X}$, P, $\mathbf{x}_E$, X_i)

Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$ in topological order. The distributions in the network $P = \{p_1, \ldots, p_N\}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$. The target variable X_i.

Output: A simulated sample and its weight.

```
1  begin
2      v1 ← 1 ; v2 ← 1.
3      for j ← 1 to N do
4          if X_j ∉ X_E then
5              Simulate a value x_j for X_j using p_j(x_j|pa(x_j)).
6              v2 ← v2 · p_j(x_j|pa(x_j)).
7          end
8          else
9              Let x_j be the value of X_j in X_E.
10         end
11         v1 ← v1 · p_j(x_j|pa(x_j)).
12     end
13     W ← v1/v2
14     return (x_i, W).
15 end
```

Algorithm 2: The EW algorithm for simulating a sample and computing its weight.

for approximating a single Gaussian. A pseudo-code description is given in Algorithm 3.

Function EW_MoG($\mathbf{X}$, P, $\mathbf{x}_E$, X_i, M)

Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$ in topological order. The distributions in the network $P = \{p_1, \ldots, p_N\}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$. The target variable X_i. Sample size M.

Output: An estimate of the parameters of the MoG distribution $p(x_i|\mathbf{x}_E)$.

begin
  $\boldsymbol{\theta}^{(0)} \leftarrow$ Random Initialization. $sumW \leftarrow 0$
  for $j \leftarrow 1$ **to** M **do**
    $(x_i^{(j)}, W) \leftarrow$ EW_Simulate($\mathbf{X}$, P, $\mathbf{x}_E$, X_i)
    If necessary, add a new component to the mixture (see Equ. (19)).
    $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)} + \rho_j \hat{\nabla}_{\boldsymbol{\theta}} \ell(x_i^{(j)}|\boldsymbol{\theta}^{(j-i)})$ (see Equ. (16) and Equ. (17))
    $sumW \leftarrow sumW + W$
  end
  return $(\boldsymbol{\theta}^{(M)}, sumW)$.
end

Algorithm 3: The EW algorithm approximating the posterior distribution $p(x_i|\boldsymbol{x}_E)$ with a MoG density, whose number of components is updated dynamically.

One might also have considered the application of the standard EM algorithm (Dempster et al., 1977) to address this problem. However, in situations with limited computing resources and where quick responses are required this approach is not feasible. The main problem is the iterative nature of the EM algorithm, which requires storing the generated samples and using them to update the parameters of the Gaussian mixture until convergence. Next, we show that it is possible to use a stochastic gradient ascent based method to fit the MoG taking as input the weights generated by importance sampling, with no need to iterate over the generated sample. Our approach can be seen as a form of online EM estimation, extending some of the online EM algoritms in the literature (see (Pinto & Engel, 2015)).

Let us start by highlighting that the minimization problem stated in Equa-

tion (11) is equivalent to the following maximization problem:

$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\big[\ln q(x_i|\boldsymbol{\theta})\big], \tag{13}$$

since

$$\begin{aligned} KL\big(p(x_i|\boldsymbol{x}_E)|q(x_i|\boldsymbol{\theta})\big) &= \int p(x_i|\boldsymbol{x}_E)\ln\frac{p(x_i|\boldsymbol{x}_E)}{q(x_i|\boldsymbol{\theta})}dx_i \\ &= \int p(x_i|\boldsymbol{x}_E)\ln p(x_i|\boldsymbol{x}_E)dx_i - \int p(x_i|\boldsymbol{x}_E)\ln q(x_i|\boldsymbol{\theta})dx_i \\ &= \mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\big[\ln p(x_i|\boldsymbol{x}_E)\big] - \mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\big[\ln q(x_i|\boldsymbol{\theta})\big], \end{aligned} \tag{14}$$

That is, the expressions in Equation (11) and Equation (13) differ by only a constant term (which corresponds to the true posterior entropy).

Note that the above maximization problem resembles a maximum likelihood problem where $p(x_i|\boldsymbol{x}_E)$ acts as the data-generating distribution. Indeed, we can rewrite Equation (13) using the importance sampling formulation as:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\big[\ln q(x_i|\boldsymbol{\theta})\big] \\ &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p^*(x_i)}\left[\frac{p(x_i|\boldsymbol{x}_E)}{p^*(x_i)}\ln q(x_i|\boldsymbol{\theta})\right] \\ &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p^*(x_i)}\left[\frac{g(x_i)}{p^*(x_i)}\ln q(x_i|\boldsymbol{\theta})\right] \\ &= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{p^*(x_i)}\big[\ell(x_i|\boldsymbol{\theta})\big]; \end{aligned} \tag{15}$$

here we replace the term $p(x_i|\boldsymbol{x}_E)$ by $g(x_i) = p(x_i, \boldsymbol{x}_E)$ because multiplication by the constant $p(\boldsymbol{x}_E)$ does not affect the maximization problem. The last equation shows that our problem reduces to a maximization problem of an expected loss function, where $\ell(x_i|\boldsymbol{\theta}) = \frac{g(x_i)}{p^*(x_i)}\ln q(x_i|\boldsymbol{\theta})$.

Stochastic gradient ascent (SGA) can be applied to solve this maximization problem. Given a Robbins-Monro's (Robbins & Monro, 1951) sequence of learning rates $\{\rho_j\}_{j\in\mathbb{N}}$ (i.e. $\rho_j > 0$, $\sum_j \rho_j = \infty$ and $\sum_j \rho_j^2 < \infty$), the SGA can be expressed as a recursive updating equation following noisy estimates of the

natural gradient (Amari, 1998) of the expected loss function

$$\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)} + \rho_j \hat{\nabla}_{\boldsymbol{\theta}} \ell(x_i^{(j)} | \boldsymbol{\theta}^{(j-1)}), \tag{16}$$

where $x_i^{(j)}$ is the j-th sample provided by the proposal distribution p^*, and $\hat{\nabla}$ denotes the natural gradient, which, unlike the standard gradient, considers the Riemannian structure of the probability space and provides faster convergence (Amari, 1998).

As shown in (Masegosa, 2014), Equation (16) can be computed in closed form. Firstly, $\boldsymbol{\theta}$ will be defined over the so-called *moment parameters* of the MoG density, which is a vector composed by $3 \cdot K$ components: the weight of each Gaussian, denoted by w_k; the mean of each Gaussian, denoted by μ_k; and the second order moment of each Gaussian, denoted by ν_k. Then, the (natural) gradient of $\ell(x_i^{(j)} | \boldsymbol{\theta}^{(j-1)})$ is computed as follows (full details are given in Appendix A):

$$\hat{\nabla}_{\boldsymbol{\theta}} \ell = \begin{pmatrix} \frac{g(x_i)}{p^*(x_i)}(q(Z=0|x_i) - \theta_0) \\ \cdots \\ \frac{g(x_i)}{p^*(x_i)}(q(Z=K-1|x_i) - \theta_{K-1}) \\ \frac{g(x_i)}{p^*(x_i)}(x_i \cdot q(Z=0|x_i) - \theta_K) \\ \cdots \\ \frac{g(x_i)}{p^*(x_i)}(x_i \cdot q(Z=K-1|x_i) - \theta_{2K-1}) \\ \frac{g(x_i)}{p^*(x_i)}(x_i^2 \cdot q(Z=0|x_i) - \theta_{2K}) \\ \cdots \\ \frac{g(x_i)}{p^*(x_i)}(x_i^2 \cdot q(Z=K-1|x_i) - \theta_{3K-1}) \end{pmatrix}, \tag{17}$$

where $q(Z = k|x_i)$ denotes the posterior probability that the observation x_i belongs to the k-th Gaussian component, $q(Z = k|x_i) \propto \mathcal{N}(x_i|\mu_k, \sigma_k^2) w_k$ (here Z is acting as a multinomial random variable and $q(Z = k|x_i)$ will be a function of x_i, different for each k)

From $\boldsymbol{\theta}$, we can compute the means, variances, and weights defining the

MoG density by using the following equality (see Appendix A):

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_0 \\ \cdots \\ \theta_{K-1} \\ \theta_K \\ \cdots \\ \theta_{2K-1} \\ \theta_{2K} \\ \cdots \\ \theta_{3K-1} \end{pmatrix} = \begin{pmatrix} w_0 \\ \cdots \\ w_{K-1} \\ w_0\mu_0 \\ \cdots \\ w_{K-1}\mu_{K-1} \\ w_0\nu_0 \\ \cdots \\ w_{K-1}\nu_{K-1} \end{pmatrix}, \tag{18}$$

where ν_k is the second order non-central moment of the k-th component (i.e. $\mathbb{E}[X^2]$ of the k-th Gaussian density).

The following result shows the convergence of Algorithm 3. We should be aware that this algorithm is a stochastic gradient ascent method, which is guaranteed to convergence after a sufficiently large number of iterations. In our case, each iteration involves the processing of a new data sample. Some technical conditions about the *smoothness* of the loss function (Bottou, 1998) needs to be invoked to prove the convergence. They are satisfied in general for (non-degenerated) CLG models.

Theorem 1. *For a valid Robbins-Monro sequence of learning rates, Algorithm 3 is guaranteed to converge to a stationary point of Equation* (11) *when* $M \to \infty$.

Proof. If the learning rates ρ_t satisfies the theorem assumptions, the stochastic gradient ascent method of Equation (16) converges to a stationary point (i.e. null natural gradient) $\boldsymbol{\theta}^\star$ of the function $\mathbb{E}_{p^*(x_i)}\left[\ell(x_i|\boldsymbol{\theta})\right]$. This stationary point will also induce a null (standard) gradient of $\mathbb{E}_{p^*(x_i)}\left[\ell(x_i|\boldsymbol{\theta})\right]$ due to the specification of the natural gradient (see Appendix A). Furthermore, $\boldsymbol{\theta}^\star$ is also a

stationary point of $\mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\left[\ln q(x_i|\boldsymbol{\theta})\right]$ due to the following equality,

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}}\mathbb{E}_{p^*(x_i)}\left[\ell(x_i|\boldsymbol{\theta})\right] &= \mathbb{E}_{p^*(x_i)}\left[\frac{g(x_i)}{p^*(x_i)}\frac{\partial}{\partial\boldsymbol{\theta}}\ln q(x_i|\boldsymbol{\theta})\right] \\ &= \int\left[p^*(x_i)\frac{g(x_i)}{p^*(x_i)}\frac{\partial}{\partial\boldsymbol{\theta}}\ln q(x_i|\boldsymbol{\theta})\right]dx_i \\ &= \frac{\partial}{\partial\boldsymbol{\theta}}\mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\left[\ln q(x_i|\boldsymbol{\theta})\right],\end{aligned}$$

Equation (14) shows that $\mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\left[\ln q(x_i|\boldsymbol{\theta})\right]$ and $KL(p(x_i|\boldsymbol{x}_E)|q(x_i|\boldsymbol{\theta}))$ differ by constant terms with respect to $\boldsymbol{\theta}$. Thus, $\boldsymbol{\theta}^\star$ is also a stationary point of $KL(p(x_i|\boldsymbol{x}_E)|q(x_i|\boldsymbol{\theta}))$. □

Under the standard assumption that local minima and saddle points are not stable convergent points of a stochastic gradient ascent algorithm (these points can easily be escaped by this method through small perturbations), we have that the stochastic gradient ascent method converges to a local minimum of $KL(p(x_i|\boldsymbol{x}_E)|q(x_i|\boldsymbol{\theta}))$. Then, Algorithm 3 provides a (local) optimal solution to the problem of approximating the true posterior $p(x_i|\boldsymbol{x}_E)$ with a MoG density.

In order to define the number of components in our MoG, we use the heuristic method described in (Pinto & Engel, 2015) to automatically decide when to create a new component in a MoG density. The criterion proposed in this paper relies on two parameters: the *novelty rate*, $0 < \tau_{\text{now}} < 1$ and the initial variance, σ^2_{ini}. Given a Gaussian mixture of K components with means μ_k and variance σ^2_k, for $k = 1, \ldots, K$, and a new data point x, the density value of x corresponding to component k is computed as:

$$p(x|k) = \frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right).$$

If it holds that

$$p(x|k) < \frac{\tau_{\text{now}}}{\sqrt{2\pi}\sigma_k}, \quad \text{for all } k = 1, \ldots, K, \tag{19}$$

a new component is added to the MoG, with mean $\mu_{K+1} = x$ and variance $\sigma^2_{K+1} = \sigma^2_{\text{ini}}$.

The novelty rate parameter represents the percentage of the maximum Gaussian density value from which the data point x will be considered as drawn from

a different component. Typical values of τ_{now} range from 0.1 to 0.0001, according to the amount of components intended to be obtained.

Function ParellelEW_normal($\mathbf{X}$, P, $\mathbf{x}_E$, X_i, M, R)

Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$ in topological order. The distributions in the network $P = \{p_1, \ldots, p_N\}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$. The target variable X_i. Sample size M. The number of parallel computing nodes R.

Output: An estimation of the mean and variance of $p(x_i|\mathbf{x}_E)$.

```
1  begin
2      sumW ← 0
3      for k ← 1 to R in parallel do
4          s_{1,k}, s_{2,k}, sumW_k = EW_normal(X, P, x_E, X_i, M/R)
5      end
6      sumW ← Σ_{k=1}^{R} sumW_k
7      s_1, s_2 ← 0
8      for k ← 1 to R do
9          s_1 ← s_1 + s_{1,k}
10         s_2 ← s_2 + s_{2,k}
11     end
12     μ ← s_1/sumW
13     σ^2 ← s_2/sumW − μ^2
14     return μ, σ^2 .
15 end
```

Algorithm 4: The Parallel EW algorithm for estimating in parallel the mean and variance of the posterior distribution.

4.3. Online parallel fitting of a Gaussian density

In regards to the scalability of importance sampling as described in Algorithm 1, it is worth pointing out that the iterations in the *for-loop* for sample generation (starting from Line 3) can be executed in parallel. This is due to the fact that the items in the sample are independent of each other. As that loop constitutes the fundamental workload of the algorithm, the opportunity

for scalability is potentially high. A straightforward proposal for scaling up the algorithm is shown in Algorithm 4.

This parallelization scheme follows a Map/Reduce approach as depicted in Figure 1, where the Map operation computes the local estimations in parallel (parallel for-loop of Line 3), and in the reduce phase the output of the map operations are combined (for-loop of Line 8).

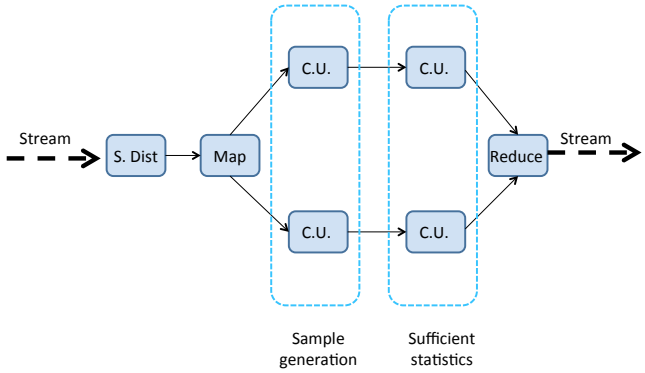


Figure 1: A Map/Reduce scheme of the design of the parallel importance sampling algorithm. Acronym c.u. stands for *computing unit* and S. Dist means computation of the sampling (proposal) distributions.

Theorem 2. *Algorithm 4 provides an unbiased estimate of the mean and the variance of the posterior distribution* $p(x_i|\boldsymbol{x}_E)$.

Proof. The above assertion can be proof by decomposing the standard unbiased importance sampling estimate of the expected value of a function f in R different groups,

$$\begin{aligned} \hat{\mathbb{E}}_p[f(X_i)] &= \frac{1}{W}\sum_{j=1}^{m}\frac{g(x^{(j)})}{p^*(x^{(j)})}f(x^{(j)}) \\ &= \sum_{r=1}^{R}\frac{1}{W}\left(\sum_{j=1}^{M/R}\frac{g(x^{(j)})}{p^*(x^{(j)})}f(x^{(j)})\right), \end{aligned} \tag{20}$$

where $W_r = \sum_{j=1}^{m^r} \frac{g(x^{(j)})}{p^*(x^{(j)})}$ and $W = \sum_{r=1}^{R} W_r = \sum_{j=1}^{m} \frac{g(x^{(j)})}{p^*(x^{(j)})}$. The quantity $\sum_{j=1}^{M/R} \frac{g(x^{(j)})}{p^*(x^{(j)})} f(x^{(j)})$ is the one computed by Algorithm 1 for $f(x) = (x, x^2)$. □

4.4. Online parallel fitting of a MoG density

In this section we present a novel parallel approach for fitting a MoG density, which again follows a Map/Reduce scheme as shown in Figure 1.

Algorithm 5 shows the pseudo-code for the proposed method. The Map operation computes the local MoGs in parallel (parallel for-loop, Line 3). In the Reduce phase (starting at Line 7) the local weights are rescaled (Line 11) by the overall weight of the local MoG component $\gamma_h = \frac{sumW_h}{sumW}$. Finally, all the MoG components are aggregated (Line 14) into a new MoG density, which contains all the local mixture components computed in parallel. Notice that K_h, the number of components obtained in each node can be different in general.

Before proving the soundness of this algorithm, we need to define a *hierarchy of MoG densities* (HMoG). A HMoG density is defined as a mixture of MoG densities. We denote by $q_{\mathcal{H}}$ a joint density defined as follows,

$$q_{\mathcal{H}}(x_i, h|\boldsymbol{\theta}, \boldsymbol{\gamma}) \quad = \quad q_{\mathcal{H}}(h|\boldsymbol{\gamma}) q_{\mathcal{H}}(x_i|h, \boldsymbol{\theta}_h) = \gamma_h q_{\mathcal{H}}(x_i|\boldsymbol{\theta}_h), \tag{21}$$

where $q_{\mathcal{H}}(x_i|\boldsymbol{\theta}_h)$ is the h-th MoG density with parameters $\boldsymbol{\theta}_h$, and γ_h is the associated weight of the h-th MoG density. H is a multinomial random variable with R different states, which will index each of the MoG components of the hierarchy.

A HMoG density can be directly transformed into a MoG density by marginalizing out H,

$$\sum_{h=1}^{R} q_{\mathcal{H}}(x_i, h|\boldsymbol{\theta}, \boldsymbol{\gamma}) = \sum_{h=1}^{R} \gamma_h q_{\mathcal{H}}(x_i|\boldsymbol{\theta}_h) = \sum_{h=1}^{R} \sum_{i=1}^{K_h} \gamma_h w_{i,h} \mathcal{N}(x_i|\mu_{i,h}, \sigma_{i,h}). \tag{22}$$

Similarly, a MoG density can be transformed to any HMoG density which satisfies the above equality. Note that the final number of components in Equation (22) is at most $R \times \max_{h=1,\ldots,R}\{K_h\}$, where each K_h is chosen using the heuristic by Pinto & Engel (2015) as described in Section 4.2.

Algorithm 5 learns in parallel a HMoG density which, once learned, is transformed into a MoG density. This parallel learning happens by considering a collection of proposal distributions, each one is executed in parallel at the available computing nodes. Notice that we use *the same proposal evidence weighting* distribution everywhere, but have a *different seed* at each computing node for the pseudo-random number generators coding the proposal distribution.

The next result shows the convergence of Algorithm 5. Again, we should consider that this algorithm is a stochastic gradient ascent method which is guaranteed to convergence after a sufficiently large number of iterations. In our case, each iteration involves the processing of a new data sample. As in Theorem 1, some technical conditions about the *smoothness* of the loss function (Bottou, 1998) needs to be invoked to prove the convergence. These are satisfied for (non-degenerated) CLG models.

Theorem 3. *For a valid Robbins-Monro sequence of learning rates, Algorithm 5 is guaranteed to converge to a stationary point of Equation* (11) *when* $M \to \infty$.

Proof. By Equations (13-14), the minimization problem of Equation (11) can be transformed into the following maximization problem,

$$(\boldsymbol{\theta}^*, \boldsymbol{\gamma}^*) \quad = \quad \arg\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} \mathbb{E}_{p(h,x_i|\boldsymbol{x}_E)} \left[\ln q_{\mathcal{H}}(h, x_i|\boldsymbol{\theta}, \boldsymbol{\gamma})\right],$$

where $q_{\mathcal{H}}$ refers to the HMoG density parametrized by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R\}$ and $\boldsymbol{\gamma}$, and $p(x_i, h|\boldsymbol{x}_E)$ is defined as $p(h, x_i|\boldsymbol{x}_E) = p(h)p(x_i|\boldsymbol{x}_E)$ with $p(h) = \frac{1}{R}$.

We now expand the above equation as follows,

$$
\begin{aligned}
(\boldsymbol{\theta}^*, \boldsymbol{\gamma}^*) &= \arg\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} \mathbb{E}_{p(h,x_i|\boldsymbol{x}_E)} \left[\ln q_{\mathcal{H}}(h, x_i|\boldsymbol{\theta}, \boldsymbol{\gamma})\right] \\
&= \arg\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} \sum_{h=1}^{R} p(h) \mathbb{E}_{p(x_i|\boldsymbol{x}_E)} \left[\ln q_{\mathcal{H}}(x_i, h|\boldsymbol{\theta}, \boldsymbol{\gamma})\right] \\
&= \arg\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} \sum_{h=1}^{R} \frac{1}{R} \mathbb{E}_{p_h^*(x_i)} \left[\frac{p(x_i|\boldsymbol{x}_E)}{p_h^*(x_i)} \ln q_{\mathcal{H}}(x_i, h|\boldsymbol{\theta}, \boldsymbol{\gamma})\right] \\
&= \arg\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} \left(\sum_{h=1}^{R} \frac{1}{R} \mathbb{E}_{p_h^*(x_i)} \left[\frac{p(x_i|\boldsymbol{x}_E)}{p_h^*(x_i)} \ln q_{\mathcal{H}}(x_i|h, \boldsymbol{\theta})\right] \right. \\
&\quad \left. + \sum_{h=1}^{R} \frac{1}{R} \mathbb{E}_{p_h^*(x_i)} \left[\frac{p(x_i|\boldsymbol{x}_E)}{p_h^*(x_i)} \ln q_{\mathcal{H}}(h|\boldsymbol{\gamma})\right] \right),
\end{aligned}
$$

where $p_h^*(x_i)$ refers to the proposal distribution used at the h-th computing node when running, in parallel, the *EW_MoG* method of Line 4.

Taking into account that $q_{\mathcal{H}}(x_i|h, \boldsymbol{\theta}) = q_{\mathcal{H}}(x_i|h, \boldsymbol{\theta}_h)$ and that $\boldsymbol{\theta}$ is not involved in the last term of the above equation, the maximization problem decomposes as the following $R + 1$ independent maximization problems,

$$
\boldsymbol{\theta}_h^* = \arg\max_{\boldsymbol{\theta}_h} \mathbb{E}_{p_h^*(x_i)} \left[\frac{p(x_i|\boldsymbol{x}_E)}{p_h^*(x_i)} \ln q_h(x_i|\boldsymbol{\theta}_h)\right], \tag{23}
$$

with $1 \leq h \leq R$, and

$$
\boldsymbol{\gamma}^* = \arg\max_{\boldsymbol{\gamma}} \sum_{h=1}^{R} \frac{1}{R} \mathbb{E}_{p_h^*(x_i)} \left[\frac{p(x_i|\boldsymbol{x}_E)}{p_h^*(x_i)} \ln \gamma_r\right]. \tag{24}
$$

Consequently, the R different maximization problems of Equation (23) can be solved in parallel, justifying the soundness of the parallel for-loop of Line 3. Furthermore, by Theorem 1, EW_MoG provides stationary points for each of these maximization problems.

The maximization problem of Equation (24) can be solved in closed form. First we multiply by the constant $p(\boldsymbol{x}_E)$, which does not affect the solution, and regroup terms obtaining

$$
\boldsymbol{\gamma}^* = \arg\max_{\boldsymbol{\gamma}} \sum_{h=1}^{R} \frac{1}{R} \mathbb{E}_{p_h^*(x_i)} \left[\frac{g(x_i)}{p_h^*(x_i)}\right] \ln \gamma_h. \tag{25}
$$

The above maximization problem is equivalent to maximum likelihood estimation of the parameters of a multinomial variable using the pseudo-counts $\left\{ \frac{1}{R} \mathbb{E}_{p_h^*(x)} \left[\frac{g(x)}{p_h^*(x)} \right] \right\}$, so the solution is given by

$$\gamma_h^* = \frac{\mathbb{E}_{p_h^*(x_i)} \left[\frac{g(x_i)}{p_h^*(x_i)} \right]}{\sum_{s=1}^{R} \mathbb{E}_{p_s^*(x_i)} \left[\frac{g(x_i)}{p_s^*(x_i)} \right]}. \tag{26}$$

This equation can be rewritten as $\gamma_h^* = \frac{sumW_h}{sumW}$ using the notation employed in Line 9, which shows the soundness of this step of the algorithm. □

Again, we can assume that saddle points are not stable convergent points of a stochastic gradient ascent algorithm, as they are easily escaped through small perturbations. In consequence, the stochastic gradient ascent method converges to a local minimum of $KL(p(x_i|\boldsymbol{x}_E)|q(x_i|\boldsymbol{\theta}))$. Then, Algorithm 5 provides a (locally) optimal solution to the problem of approximating the true posterior $p(x_i|\boldsymbol{x}_E)$ with a HMoG density.

By considering Lines 9 and 11 in Algorithm 5 it is clear that γ_h defines how much the h-th MoG component contributes to the final MoG density. If $\gamma_h = 1$, the h-th MoG component will be the only one contributing, while the contribution will be zero if $\gamma_h = 0$. The actual contribution depends of the samples being generated since $sumW_h$ (as defined in Line 4) is equal to

$$sumW_h = \frac{1}{(M/R)} \sum_{j=1}^{M/R} \frac{g(x_i^{(j)})}{p_h^*(x_i^{(j)})}.$$

The following result states an upper bound for the KL distance of the MoG density returned by Algorithm 5.

Theorem 4. *Let us denote $q_{\mathcal{H}}(x_i|\boldsymbol{\theta}_h)$ to the h-th MoG density returned by Line 4 of Algorithm 5, and let $q_{\mathcal{H}}(x_i|\boldsymbol{\theta}, \boldsymbol{\gamma})$ denote the final combined MoG density returned by Algorithm 5. We then have the following inequality,*

$$KL\left(p(x_i|x_{\boldsymbol{E}})||q_{\mathcal{H}}(x_i|\boldsymbol{\theta}, \boldsymbol{\gamma})\right) \leq \sum_{h=1}^{R} \gamma_h KL(p(x_i|x_{\boldsymbol{E}})||q_{\mathcal{H}}(x_i|\boldsymbol{\theta}_h))$$

Proof. By Jensen's bound we have that,

$$-\ln q_{\mathcal{H}}(x_i|\boldsymbol{\theta},\boldsymbol{\gamma}) \leq -\sum_{h=1}^{R} \gamma_h \ln q_{\mathcal{H}}(x_i|\boldsymbol{\theta}_h)$$

By taking expectation wrt $p(x_i|x_{\boldsymbol{E}})$ and subtracting the entropy of $p(x_i|x_{\boldsymbol{E}})$ at both sides of the inequality, we obtain the above result. □

Theorem 4 guarantees that using a HMoG is never worse than using a single MoG in terms of *KL* divergence, and that it can be beneficial.

Function ParellelEW_MoG($\mathbf{X}$, P, $\mathbf{x}_E$, X_i, M, R)

Input: The set of variables in the network, $\mathbf{X} = \{X_1, \ldots, X_N\}$ in topological order. The distributions in the network $P = \{p_1, \ldots, p_N\}$. Evidence $\mathbf{X}_E = \mathbf{x}_E$. The target variable X_i. Sample size M. The number of parallel computing nodes R.

Output: An estimate of the parameters of the MoG distribution $p(x_i|\mathbf{x}_E)$.

begin
  $sumW \leftarrow 0$
  for $h \leftarrow 1$ **to** R ***in parallel*** **do**
    $(\boldsymbol{\theta}_h, sumW_h) = \text{EW\_MoG}(\mathbf{X}, P, \mathbf{x}_E, X_i, \frac{M}{R})$ (the result is a MoG of K_h components)
  end
  $sumW \leftarrow \sum_{h=1}^{R} sumW_h$
  for $h \leftarrow 1$ **to** R **do**
    $(\boldsymbol{\mu}_h, \boldsymbol{\sigma}_h^2, \boldsymbol{w}_h) \leftarrow \boldsymbol{\theta}_h$ (following Equation (18))
    $\gamma_h \leftarrow \frac{sumW_h}{sumW}$
    for $k \leftarrow 1$ **to** K_h **do**
      $\boldsymbol{w}_h[k] \leftarrow \boldsymbol{w}_h[k] \cdot \gamma_h$
    end
  end
  return $(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_R, \boldsymbol{\sigma}_1^2, \ldots, \boldsymbol{\sigma}_R^2, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_R)$.
end

Algorithm 5: The Parallel EW algorithm for approximating the posterior distribution $p(x_i|\boldsymbol{x}_E)$ with a MoG density.

5. Experiments

We have performed several experiments in order to assess the performance of the algorithms developed in the previous section, both in terms of the accuracy when representing the posterior density of a continuous variable and with respect to their scalability, i.e, their ability to exploit the computational resources that are available.

The experiments were performed using the AMIDST toolbox (Masegosa et al. (2017b)), which is built on top of the Apache Flink framework for distributed computation, and were carried out on a dual-processor AMD Opteron 2.8GHz server with 32 cores and 64GB of RAM, running Linux Ubuntu 14.04.1 LTS. The code is available at the AMIDST GitHub repository (`https://github.com/amidst/toolbox/tree/MAP-Flink`).

5.1. Toy example

We first start the experimental section with a toy example, using it as proof of concept while also emphasizing the necessity of introducing Gaussian mixtures for representing the posterior distribution in a CLG model. Consider a hidden Markov model (HMM), which is indeed a CLG dynamic Bayesian network, with T time steps in total. At each time step $t = 1, \ldots, T$, the model consists of a discrete hidden variable X_t being the parent of an observable continuous variable Y_t. The hidden variables $\{X_1, \ldots, X_T\}$ are also connected over time, so that X_t is a parent of X_{t+1}. We will be interested in the posterior density of the last continuous variable, Y_T, possibly with some known evidence on the values of its previous temporal copies. The true posterior density of Y_T is a MoG, whose number of components is the same as the number of states of the hidden variable X_T.

The posterior distribution of Y_T for a sample HMM where X_T has 4 states is depicted in Fig. 2 (red line). This plot also includes the estimated posterior densities found by the algorithms that we propose, both using a Gaussian density (blue line) and a MoG (green line). Even though the estimated MoG does not

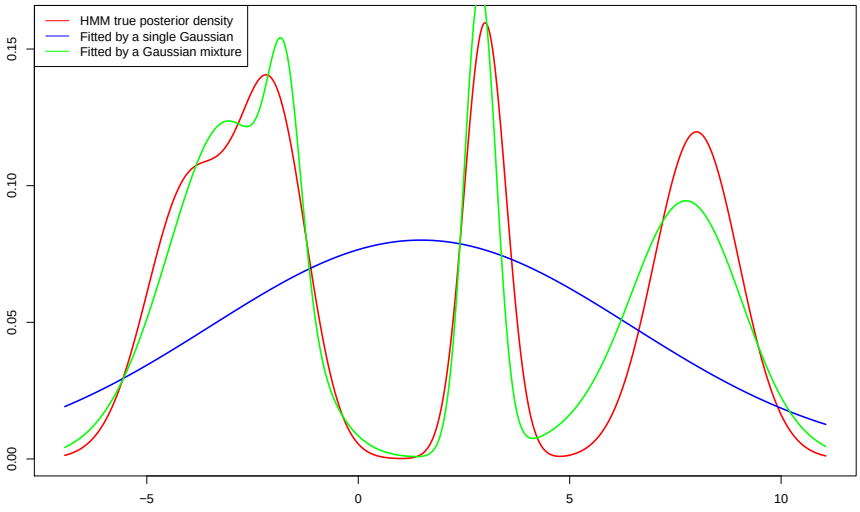


Figure 2: True posterior density of a HMM (red) and estimated posterior densities fitted by a single Gaussian (blue) and a Gaussian mixture (green), according to the proposed procedure.

match the true posterior density perfectly, it is much more accurate than the single Gaussian estimate. The use of a single Gaussian representation for this distribution leads to remarkable imprecisions, as there are regions where the true density has almost no probability mass while the single Gaussian estimate still assigns high density values to those regions.

This is a typical situation in many models, which justifies the use of MoGs for approximating the posterior densities, instead of using a single Gaussian as many approximate inference methods (Minka, 2001; Winn & Bishop, 2005).

5.2. Accuracy assessment

We have tested the algorithms with two dynamic models that were discussed in (Ramos-López et al., 2017); both of them are dynamic CLG Bayesian networks. We employ dynamic models because they constitute a relevant and challenging case of CLG models and, also, because they are commonly used for processing data streams. We give a brief description of these models, and refer

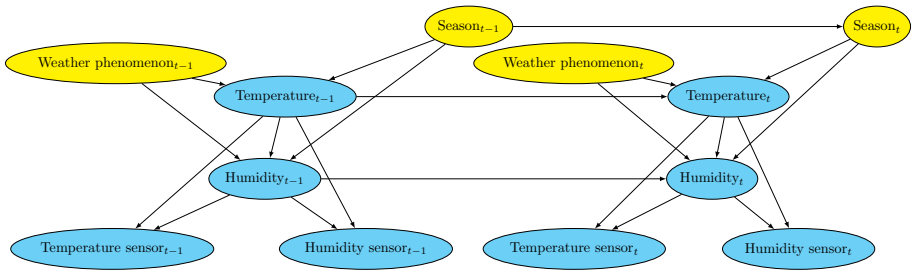


Figure 3: DAG of the dynamic Bayesian network model used for detecting season changes (yellow nodes correspond to discrete variables; cyan nodes correspond to continuous variables; the observable nodes are 'Temperature sensor' and 'Humidity sensor').

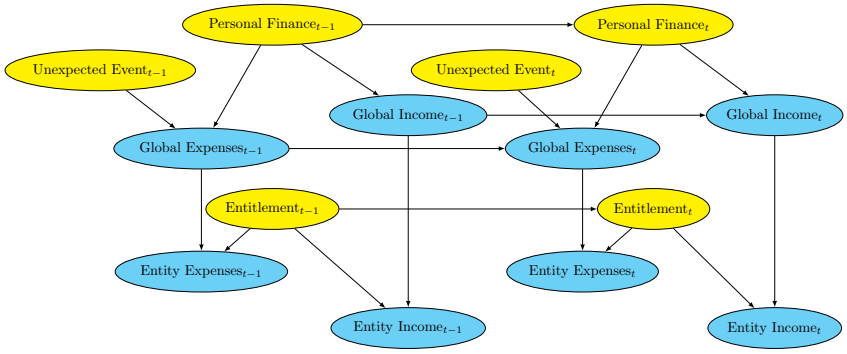


Figure 4: DAG of the dynamic Bayesian network model used for detecting changes in personal finances (yellow nodes correspond to discrete variables; cyan nodes correspond to continuous variables; the observable nodes are 'Entity expenses' and 'Entity income').

the readers to (Ramos-López et al., 2017) for further information.

The first model ("Season change"), consist of a hidden layer of two discrete variables ('Season' and 'Weather phenomenon'), and two continuous variables ('Temperature' and 'Humidity'), and an observable layer of two continuous variables ('Temperature sensor' and 'Humidity sensor'). The network structure is shown in Fig. 3.

The second model ("Personal finances"), also includes a hidden layer, now with three discrete nodes ('Personal finance', 'Unexpected event' and 'Entitlement') and two continuous variables ('Global expenses' and 'Global income'). The observable layer consists of two continuous variables ('Entity expenses' and

'Entity income'). The network structure of this model is shown in Fig. 4.

Two different experiments have been carried out with each model. Each of the two dynamic models is unrolled to T time steps (we will say T is the network length), and evidence is given for the observable variables at times $t = 1, \ldots, T-1$, whereas the variable of interest will be a continuous node in the last time slice ('Temperature' for the season change model and 'Global income' for the personal finance model). Then, the posterior distribution of that variable is estimated using the two algorithms based on importance sampling: with a single Gaussian, and with a MoG. For comparison the distribution has also been estimated with the variational message passing (VMP) algorithm (Winn & Bishop, 2005). As a side-note we remark that while several recent works, like *black-box variational inference* (Ranganath et al., 2014), study the use of variational inference in broad model classes, our interest here is to analyze a model within the conjugate exponential family of distributions. For models in this distributional family it is known that the VMP algorithm performs optimal parameter updates, hence will perform at least as good as any other algorithm using the variational objective (Winn & Bishop, 2005). As a consequence, we do not compare our approach to other variational inference algorithms; neither of these alternatives will be able to outperform VMP.

We first consider a network with fixed length $T = 8$. The posterior density of the variable of interest was estimated by each method with an increasing sample size (ranging from $10^{3.5}$ to 10^6). Finally, to measure the accuracy of each of the algorithms, the average log-likelihood of each posterior density was computed, simulating an independent sample of size 10^6. This quantity is closely related to the term $KL(p(x_i|\boldsymbol{x}_E)||q(x_i|\boldsymbol{\theta}))$ in the minimization problem in Equation (11) (the KL distance is a constant term minus this log-likelihood, see Equation (14)). The experiment has been repeated 20 times with changing evidence.

The results are plotted in Figure 5 for the season change model and Figure 6 for the personal finances model. The top row of each figure shows the average log-likelihood across the 20 repetitions, whereas the bottom rows give the distributions of values summarized by box-plots (excluding extreme outliers).

The VMP log-likelihood mean value is constant, as its estimated density does not depend on the sample size, only on the evidence. Note that the scale of the VMP results (right axis, top row of Figures 5 and 6) is much larger than that of the importance sampling based algorithms[2], and for this reason this method has not been included in the box-plots below. The plots show increasing accuracy of the posterior density estimates when using importance sampling. They also demonstrate that the MoG estimates perform much better than the single Gaussian alternative. According to the results, the precision of the single Gaussian scheme is more variable than the MoG version, as the box-plots are much wider in general. This makes sense, as the dependence of the posterior distribution on the observed evidence can be dramatic. Although all the posteriors are MoGs, some of them may be strongly dominated by a single Gaussian, so that it could be well approximated by a single Gaussian. On the other hand, when the true posterior is a MoG with balanced weights, a single Gaussian may be unable to fit it properly, as was also evident in our first experiment, cf. Figure 2.

Next we consider Bayesian networks of increasing size, while keeping the number of samples used in the importance sampling fixed at 10^6. With a network length ranging from $T = 3$ to $T = 18$, and given the evidence, the posterior density of the variable of interest was estimated by each method. Again, the experiment was repeated 20 times for each network size, and the log-likelihood of each density was computed. The results are shown in Figures 7 and 8. The results are consistent with those obtained in the previous experiment. VMP obtains a log-likelihood that is much smaller than the importance sampling algorithms (see the top row of Figures 7 and 8, and compare the values on the left and right axes). The plots show how the accuracy of the single Gaussian estimate drops when the network length grows, while the MoG results are more

[2]The poor VMP performance is due to the underestimation of the posterior variance. This characterizes approaches based on the variational objective, and is particularly prominent in time-series models (Turner & Sahani, 2011).

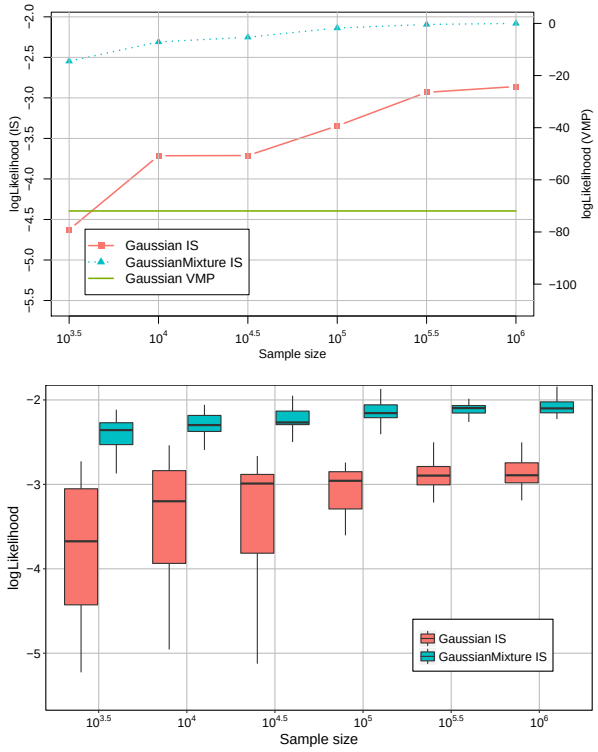


Figure 5: Log-likelihood of the estimated posterior densities with $T = 8$ and increasing sample size for the season change model. Top: mean log-likelihood value; bottom: distribution of log-likelihood values (excluding outliers).

stable for both models. The variability of the results seems to grow faster in the season change model than in the personal finances model, but for both models we observe that the behavior of the MoG density estimate is more consistent than that of the single Gaussian.

5.3. Scalability

Additional experiments were made to analyze the scalability of the proposed approaches. The VMP algorithm has been excluded from these analysis, as it basically is sequential. The experimental setup was as follows: A BN of 25000 variables was randomly generated, in which half of the variables are discrete, and the other half continuous. The number of links was set to 37500. Then,

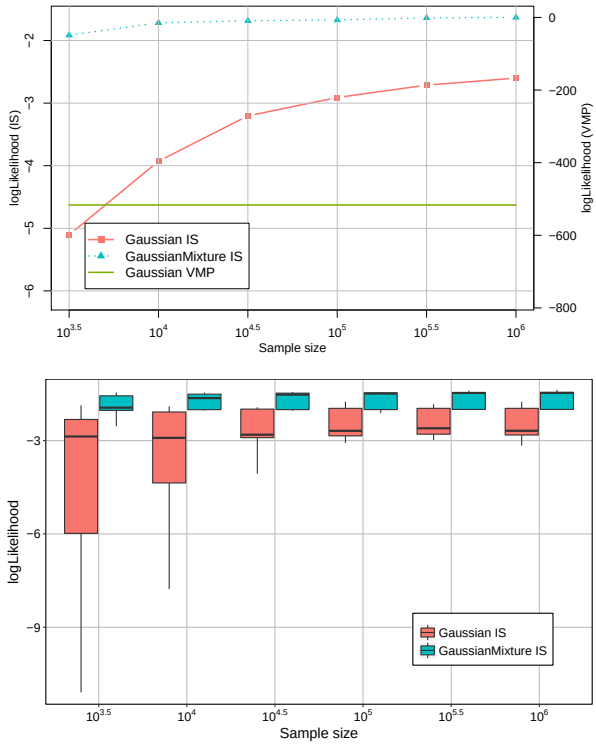


Figure 6: Log-likelihood of the estimated posterior densities with $T = 8$ and increasing sample size for the personal finances model. Top: mean log-likelihood value; bottom: distribution of log-likelihood values (excluding outliers).

evidence was introduced to 20% of the variables, and 10% of the non-observed variables were chosen as the variables of interest. Inference was carried out over the variables of interest, and their densities were estimated either by a single Gaussian or by a MoG, using a sample size of 10000. The execution time for each scheme was measured by averaging over 10 repetitions, changing the BN and the evidence between runs. Run-times were obtained using 1, 2, 4, 8, 16, 24, and 32 cores. Figure 9 depicts the obtained execution times (top, note the log-scale on the axes) and their corresponding speed-up factor (bottom) for the two alternatives. In each case, the scale-up factor for n cores is computed as the execution time of that method using 1 core divided by the execution time with n cores.

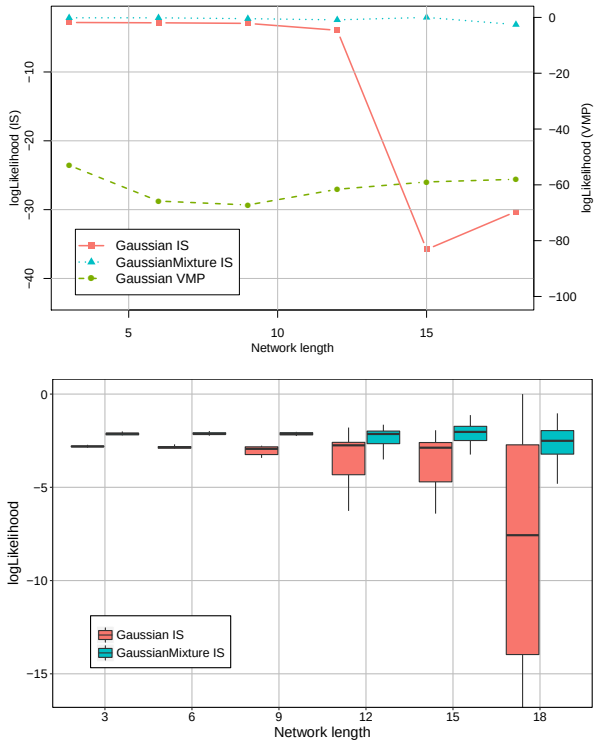


Figure 7: Mean log-likelihood of the estimated posterior densities with a fixed sample size and increasing T for the season change model. Top: mean log-likelihood value; bottom: distribution of log-likelihood values (excluding outliers).

Figure 9 demonstrates the scalability of the proposed algorithms, as we observe that the speed-up is consistently increasing with the number of cores made available. We note that our implementation is built on top of Apache Flink, which was run locally. We hypothesize that this design choice has introduced an extra computational burden that is particularly evident for the run-times using 1 or 2 cores, and which therefore has lead to surprisingly large speed-up factors for some configurations (speed-up larger than n when running on n cores, $4 \leq n \leq 16$). For the largest configurations ($n \geq 24$) we observe diminishing return from additional cores, which we attribute to the relatively small sampling workload (since the number of samples is not very large) with respect to the overhead produced by Flink, preventing the extra cores from being utilized at

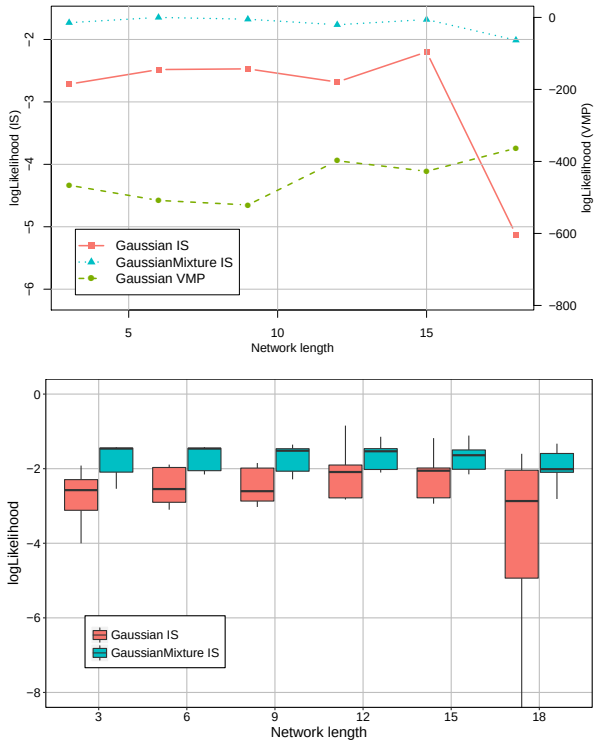


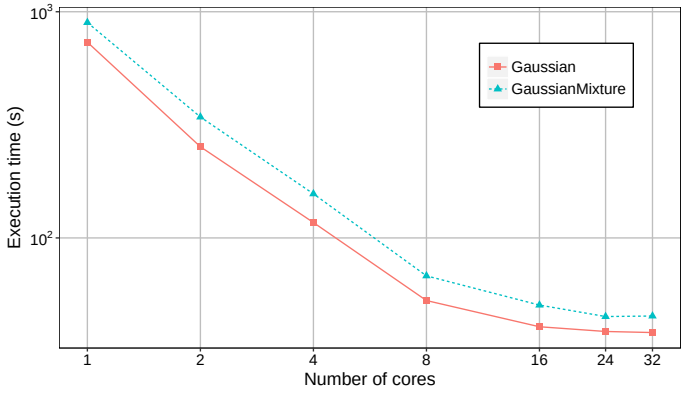
Figure 8: Mean log-likelihood of the estimated posterior densities with a fixed sample size and increasing T for the personal finances model. Top: mean log-likelihood value; bottom: distribution of log-likelihood values (excluding outliers).

their maximum capacity.

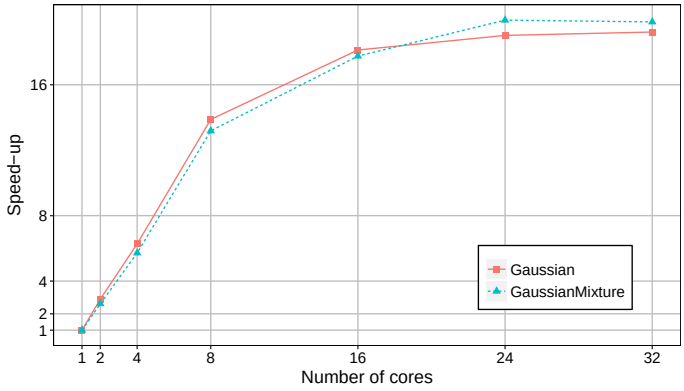
The results for different BN sizes and number of samples show a consistent behavior, yielding plots that are very similar to those in Figure 9. This makes sense since the execution time is approximately proportional to the total number of samples drawn from all the variables (the number of unobserved variables times the sample size), for any number of cores.

6. Conclusions

We have described a scalable importance sampling algorithm for computing MoG posteriors in CLG networks. Our contribution is based on using a



(a) Execution times



(b) Speed-up factor

Figure 9: Execution times and speed-up of the two importance sampling alternatives in a BN with 25000 variables and using 10000 samples. Each point represents the average value among 10 repetitions.

stochastic gradient ascent procedure that operates over the importance sampling weights in such a way that the parameters of the MoG density are updated in an on-line fashion with no need to store the full sample. The algorithm has been designed following a Map/Reduce approach and is therefore scalable with respect to computing resources. The provided theoretical background justifies both the parallel computation of the posterior densities (Map stage) and their eventual combination (Reduce stage), for a single Gaussian and a MoG density.

This procedure could easily be extended to multivariate MoG densities, by changing the sufficient statistics of the posterior distributions of interest.

The experiments carried out show that the proposed method based on MoGs outperforms the single-Gaussian and variational alternatives in terms of accuracy. Also, the empirical results show a good performance in terms of scalability.

Acknowledgements

This work was partly carried out as part of the AMIDST project. AMIDST has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619209. This research has been partly funded by the Spanish Ministry of Economy and Competitiveness, through projects TIN2013-46638-C3-1-P, TIN2015-74368-JIN, TIN2016-77902-C3-3-P and by ERDF funds. DRL, AM, AS and RR thank the support from CDTIME. DRL thanks also to CEIMAR.

Appendix A. Natural Gradients and MoG

A MoG density does not belong to the exponential family but if we consider a extended model including a multinomial indicator variable Z, then this extended model does belong to the exponential family,

$$q(x, z) = q(x|z)q(z),$$

where $q(x|z)$ is a Gaussian density and $q(z)$ is a multinomial distribution.

The above joint density can be expressed in exponential family form,

$$\ln q(x, z|\eta) = \eta^T s(x, z) - A(\eta) + h(x),$$

where η is the vector of natural parameters, $s(\cdot)$ is the vector of sufficient statistics, $A(\cdot)$ is the log-normalizer and $h(\cdot)$ is the base measure. In this case, the sufficient statistics vector for the joint MoG density is expressed as follows,

$$s(x,z) = \begin{pmatrix} I(z=0) \\ \cdots \\ I(z=K-1) \\ I(z=0)x \\ \cdots \\ I(z=K-1)x \\ I(z=0)x^2 \\ \cdots \\ I(z=K-1)x^2 \end{pmatrix},$$

where $I(\cdot)$ denotes the indicator function.

An exponential family distribution can be alternatively parametrized by a vector of moment parameters, denoted by $\boldsymbol{\theta}$, which is defined as follows,

$$\boldsymbol{\theta} \equiv \mathbb{E}[s(x,z)|\eta] = \int s(x,z)q(x,z|\eta)dxdz = \nabla_\eta A(\eta).$$

Therefore,

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_0 \\ \cdots \\ \theta_{K-1} \\ \theta_K \\ \cdots \\ \theta_{2K-1} \\ \theta_{2K} \\ \cdots \\ \theta_{3K-1} \end{pmatrix} = \begin{pmatrix} \mathbb{E}[I(z=0)] \\ \cdots \\ \mathbb{E}[I(z=K-1)] \\ \mathbb{E}[I(z=0)x] \\ \cdots \\ \mathbb{E}[I(z=K-1)x] \\ \mathbb{E}[I(z=0)x^2] \\ \cdots \\ \mathbb{E}[I(z=K-1)x^2] \end{pmatrix} = \begin{pmatrix} w_0 \\ \cdots \\ w_{K-1} \\ w_0\mu_0 \\ \cdots \\ w_{K-1}\mu_{K-1} \\ w_0\nu_0 \\ \cdots \\ w_{K-1}\nu_{K-1} \end{pmatrix},$$

where ν_k denotes the second moment, $\eta_k = \mathbb{E}[x^2]$ of the k-th Gaussian component. From the above equation we can see how to get the mean and the variance of each of the components of the MoG model, e.g. $w_0 = \theta_0$, $\mu_0 = \theta_K/\theta_0$ and $\sigma_0^2 = \theta_{2K-1}/\theta_0 - (\theta_K/\theta_0)^2$.

The natural gradient (Amari, 1998) of a loss function ℓ is defined as follows,

$$\hat{\nabla}_{\boldsymbol{\theta}}\ell = F(\boldsymbol{\theta})^{-1}\nabla_{\boldsymbol{\theta}}\ell,$$

where $F(\cdot)$ is the Fisher information matrix and ∇ denotes the standard gradient.

As shown in (Masegosa, 2014, Theorem 2), the natural gradient of a loss function w.r.t. the moment parameters equals the gradient with respect to the natural parameters,

$$\hat{\nabla}_{\boldsymbol{\theta}} \ell = \nabla_{\eta} \ell \tag{A.1}$$

Then, the (natural) gradient detailed in Equation (17) is derived as follows,

$$\begin{aligned}
\hat{\nabla}_{\boldsymbol{\theta}} \ell(x_i|\boldsymbol{\theta}) = \nabla_{\eta} \ell(x_i|\eta) &= \frac{g(x_i)}{p^*(x_i)} \nabla_{\eta} \ln \sum_z q(x_i, z|\eta) \\
&= \frac{g(x_i)}{p^*(x_i)} \frac{\sum_z \nabla_{\eta} q(x_i, z|\eta)}{q(x_i|\eta)} \\
&= \frac{g(x_i)}{p^*(x_i)} \frac{\sum_z (s(x_i, z) - \nabla_{\eta} A(\eta)) q(x_i, z|\eta)}{q(x_i|\eta)} \\
&= \frac{g(x_i)}{p^*(x_i)} \left(\sum_z (s(x_i, z) q(x_i|z, \eta)) - \boldsymbol{\theta} \right),
\end{aligned}$$

where we have used that the importance sampling weights do not depend on $\boldsymbol{\theta}$ and Equation (A.1). The gradient shown in Equation (17) is directly obtained by inspecting each component of the sufficient statistics vector.

Appendix B. Numerical instability management

In spite of the sound theoretical background of the importance sampling approach, in the implementation of the simulation process there are a number of steps in which numerical instability errors might arise (especially due to underflow). A proper treatment of these issues is mandatory in order to obtain accurate and robust estimations. Here we explain some of the mathematical tricks that allows us to deal with those numerical problems.

One of the main problems is that the importance sampling weights given by $w = v_1/v_2$ (with v_1, v_2 as defined in Algorithm 2), are likely to underflow to 0, when the number of variables is large and/or when $p(\mathbf{x}_E)$ is close to

zero. To avoid this, a possible solution is to do the calculations by employing their logarithms $\ln w_1$ and $\ln w_2$, instead of w_1 and w_2 themselves. These logarithms can be computed using the log-probability of each conditional distribution, $\ln p_i(x_i|pa(x_i))$, that are then summed up instead of multiplied (see Lines 6 and 11 of Algorithm 1).

Now, the problem is that we need to sum these quantities, alone or multiplied by other numbers (see Lines 5 to 7 of Algorithm 1). But expressed as logarithms, we cannot sum them up directly. We will use the following expression, which is often called the *log-sum-exp trick*. Let us assume that $a, b > 0$ and $a > b$, then:

$$\ln(a+b) = \ln\left(a\Big(1+\frac{b}{a}\Big)\right) = \ln a + \ln\left(1+\frac{b}{a}\right) =$$
$$= \ln a + \ln\left(1+e^{\ln b - \ln a}\right).$$

Similarly:

$$\ln(a-b) = \ln a + \ln\left(1-e^{\ln b - \ln a}\right).$$

These expressions, along with the properties of the logarithm, allow the computation of the numerical value of the logarithm of the sum (or difference) of two numbers from their logarithms, with no need to explicitly compute the numbers. In addition, many programming languages include accurate functions to compute $\ln(1 \pm x)$ when x is small, as it is in our case (since $a > b$). These functions are often named *log1p*, or something similar.

These tricks can be employed in Algorithm 1 easily. The main problem appears in the calculation of the weighted sum of the samples, in Line 5. As $x^{(j)}$ can possibly be negative and we need to compute its logarithm, we have to split this in two different sums, one of terms $\ln x^{(j)} + \ln(v_1/v_2)$ for $x^{(j)} > 0$ and another with terms $\ln(-x^{(j)}) + \ln(v_1/v_2)$ for $x^{(j)} < 0$. At the end of the procedure, these two partial sums are combined, taking into account the sign of the one with a larger absolute value. For the weighted sum of squares (Line 6), there is no problem as this value is always non-negative.

The tricks above can be adapted and extended for their use in Algorithms 3, 4 and 5 in a similar way. The logarithmic representation of the weights permits the application of the importance sampling scheme to (almost) arbitrarily large networks, or when the evidence is extremely unlikely a priori (as long as the logarithms of the sample weights are computationally representable, i.e., $\ln w \neq -\infty$ for the computer).

References

Amari, S. I. (1998). Natural gradient works efficiently in learning. *Neural computation*, *10*, 251–276.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2016). Variational inference: a review for statisticians. arXiv preprint arXiv:1601.00670.

Bottou, L. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, *17*, 142.

Cheng, J., & Druzdzel, M. J. (2000). AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, *13*, 155–188.

Dempster, A. P., Larid, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Fernández, A., Rumí, R., & Salmerón, A. (2012). Answering queries in hybrid Bayesian networks using importance sampling. *Decision Support Systems*, *53*, 580–590.

Fung, R., & Chang, K. C. (1990). Weighting and integrating evidence for stochastic simulation in Bayesian networks. In M. Henrion, R. Shachter, L. Kanal, & J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence* (pp. 209–220). North-Holland (Amsterdam) volume 5.

Hammersley, J. M., & Handscomb, D. C. (1964). *Monte Carlo Methods*. Chapman & Hall.

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, *14*, 1303–1347.

Jensen, F. V., Lauritzen, S. L., & Olesen, K. G. (1990). Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, *4*, 269–282.

Jensen, F. V., & Nielsen, T. D. (2007). *Bayesian Networks and Decision Graphs*. Springer.

Kozlov, D., & Koller, D. (1997). Nonuniform dynamic discretization in hybrid networks. In D. Geiger, & P. P. Shenoy (Eds.), *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence* (pp. 302–313). Morgan & Kaufmann.

Kullbach, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*, 79–86.

Langseth, H., Nielsen, T. D., Rumí, R., & Salmerón, A. (2012). Inference in hybrid Bayesian networks with mixtures of truncated basis functions. In *Proceedings of the Sixth European Workshop on Probabilistic Graphical Models (PGM'2012)* (pp. 171–178).

Lauritzen, S. L., & Jensen, F. (2001). Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, *11*, 191–203.

Lauritzen, S. L., & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, *17*, 31–57.

Madsen, A. L. (2010). Improvements to message computation in lazy propagation. *International Journal of Approximate Reasoning*, *51*, 499–514.

Madsen, A. L., & Jensen, F. V. (1999). Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, *113*, 203–245.

Masegosa, A. R. (2014). Stochastic discriminative EM. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence* (pp. 573–582). AUAI Press.

Masegosa, A. R., Martínez, A. M., Langseth, H., Nielsen, T. D., Salmerón, A., Ramos-López, D., & Madsen, A. L. (2017a). Scaling up Bayesian variational inference using distributed computing clusters. *International Journal of Approximate Reasoning*, *8*, 435—451.

Masegosa, A. R., Martínez, A. M., Ramos-López, D., Cabañas, R., Salmerón, A., Nielsen, T. D., Langseth, H., & Madsen, A. L. (2017b). AMIDST: a Java toolbox for scalable probabilistic machine learning. *arXiv preprint arXiv:1704.01427*, .

Minka, T. P. (2001). Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)* (pp. 362–369).

Moral, S., & Salmerón, A. (2005). Dynamic importance sampling in Bayesian networks based on probability trees. *International Journal of Approximate Reasoning*, *38*, 245 – 261.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Neil, M., Tailor, M., & Marquez, D. (2007). Inference in Bayesian networks using dynamic discretisation. *Statistics and Computing*, *17*, 219–233.

Neil, M., Tailor, M., Marquez, D., Fenton, N., & Hearty, P. (2008). Modelling dependable systems using hybrid Bayesian networks. *Reliability Engineering and System Safety*, *93*, 933–939.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA.: Morgan Kaufmann Publishers Inc.

Pinto, R. C., & Engel, P. M. (2015). A fast incremental Gaussian mixture model. *Plos One*, *10 (10)*, e0139931.

Ramos, F. T., & Cozman, F. G. (2005). Anytime anyspace probabilistic inference. *International Journal of Approximate Reasoning*, *38*, 53 – 80.

Ramos-López, D., Masegosa, A. R., Martínez, A. M., Salmerón, A., Nielsen, T. D., Langseth, H., & Madsen, A. L. (2017). MAP inference in dynamic hybrid Bayesian networks. *Progress in Artificial Intelligence*, *6(2)*, 133—144.

Ranganath, R., Gerrish, S., & Blei, D. (2014). Black box variational inference. In *AISTATS* (pp. 814–822).

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Rumí, R., & Salmerón, A. (2007). Approximate probability propagation with mixtures of truncated exponentials. *International Journal of Approximate Reasoning*, *45*, 191–210.

Salmerón, A., Ramos-López, D., Borchani, H., Masegosa, A. R., Fernández, A., Langseth, H., Madsen, A. L., & Nielsen, T. D. (2015). Parallel importance sampling in conditional linear Gaussian networks. *CAEPIA'2015. Lecture Notes in Artificial Intelligence*, *9422*, 36–46.

Shachter, R. D., & Kenley, C. (1989). Gaussian influence diagrams. *Management Science*, *35*, 527–550.

Shenoy, P. P. (1997). Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, *17*, 239–263.

Shenoy, P. P., Rumí, R., & Salmerón, A. (2015). Practical aspects of solving hybrid Bayesian networks containing deterministic conditionals. *International Journal of Intelligent Systems*, *30*, 265–291.

Shenoy, P. P., & Shafer, G. (1990). Axioms for probability and belief function propagation. In R. D. Shachter, T. S. Levitt, J. F. Lemmer, & L. N. Kanal (Eds.), *Uncertainty in Artificial Intelligence 4* (pp. 169–198). North Holland, Amsterdam.

Turner, R. E., & Sahani, M. (2011). Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, & S. Chiappa (Eds.), *Bayesian Time series models* chapter 5. (pp. 109–130). Cambridge University Press.

Winn, J. M., & Bishop, C. M. (2005). Variational message passing. *Journal of Machine Learning Research*, *6*, 661–694.

Yuan, C., & Druzdzel, M. J. (2005). Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modeling*, *43*, 1189–1207.

Yuan, C., & Druzdzel, M. J. (2007). Importance sampling for general hybrid Bayesian networks. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics* (pp. 652–659).

Zhang, N., & Poole, D. (1996). Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, *5*, 301–328.